

## Review

This work describes the recognition of human activity based on the interaction between people and objects in domestic settings, specifically in a kitchen. In order to achieve the aim of recognizing activity it is necessary to establish a procedure and essential equipment. Regarding the procedure, in a simplified manner, it is based on capturing local images where the activity takes place using a colour camera (RGB), and processing the above mentioned images to recognize the present objects and its location. The interaction with the objects is classified as five types of possible actions (unchanged, add, remove, move and Indeterminate), which are used to analyze the probability of the human activity that is being performed at the moment.

As for the technological tools employed, the system works with Ubuntu as general Operating System, ROS (Robot Operating System) as framework, OpenCV (Open Source Computer Vision) for the vision algorithms used, and Python programming language.

The development starts with the segmentation using the "difference image" method that obtains the area that the objects take up in the image the recognition of objects is carried out by distinguishing them according to its colour histogram. the positioning is obtained through its centroid, applying the corresponding homography to go from the coordinate system of the image to the coordinates of the real world using comparisons of the historical and the new information of the objects we determine the actions that have been fulfilled as final stage, we filter the relevant objects on the basis of the actions carried out and compare with the objects defined for the accomplishment of every activity the result is the probability of executing each activity.



# Summary

<b>REVIEW</b>	<b>1</b>
<b>SUMMARY</b>	<b>3</b>
<b>1. GLOSSARY</b>	<b>5</b>
<b>2. PREFACE</b>	<b>7</b>
2.1. Origin of the project.....	7
2.2. Motivation .....	7
2.3. Related works.....	8
<b>3. INTRODUCTION</b>	<b>11</b>
3.1. Objectives of the project.....	11
3.1.1. General objective: .....	11
3.1.2. Specific objectives: .....	11
3.2. Scope of the project .....	11
<b>4. CONCEPTUALIZATION OF AN OBJECT</b>	<b>12</b>
4.1. Definition of an object.....	12
4.2. Requirements .....	13
4.2.1. Hardware .....	13
4.2.2. Software.....	14
4.3. Motion detector .....	15
4.4. Object recognition .....	17
4.5. Object position.....	22
4.6. Object action definition.....	24
<b>5. HUMAN ACTIVITY RECOGNITION</b>	<b>26</b>
5.1. Definition of an activity .....	27
5.2. Distance function.....	28
5.3. Results .....	31
<b>CONCLUSIONS</b>	<b>37</b>
<b>THANKS</b>	<b>39</b>
<b>BIBLIOGRAPHY</b>	<b>40</b>
Bibliographic references .....	40
Complementary bibliography.....	41



# 1. Glossary

## *BIN:*

Bin numbers. These numbers represent the intervals that you want the Histogram tool to use for measuring the input data in the data analysis. (<http://office.microsoft.com/en-us/excel-help/present-your-data-in-a-histogram-HA010238252.aspx>).

## *HOMOGRAPHY:*

In computer vision, we define planar homography as a projective mapping from one plane to another. Thus, the mapping of points on a two-dimensional planar surface to the imager of our camera is an example of planar homography. [5]

## *IBEC:*

The Institute for Bioengineering of Catalonia (IBEC) is an interdisciplinary research centre focused on bioengineering and nanomedicine, based in Barcelona. (<http://www.ibecbarcelona.eu/IBEC/about-us.html>)

## *InHANDS:*

Interactive Robotics for Human Assistance in Domestic Scenarios. (<http://inhandsproject.wordpress.com/>)

## *KINECT:*

The Kinect is a device that has two cameras and one laser-based IR projector. Each lens is associated with a camera or a projector. ([http://wiki.ros.org/kinect\\_calibration/technical](http://wiki.ros.org/kinect_calibration/technical))

## *MESSAGES:*

A message is a simple data structure, comprising typed fields. Standard primitive types (integer, floating point, Boolean, etc.) are supported, as are arrays of primitive types. Messages can include arbitrarily nested structures and arrays (much like C structs). (<http://wiki.ros.org/Messages>)

## *NODES:*

A node really isn't much more than an executable file within a ROS package. ROS nodes use a ROS client library to communicate with other nodes. Nodes can publish or subscribe to a Topic. Nodes can also provide or use a Service. (<http://wiki.ros.org/ROS/Tutorials/UnderstandingNodes#Nodes>)

**OPENCV:**

OpenCV (Open Source Computer Vision Library: is an open-source BSD-licensed library that includes several hundreds of computer vision algorithms. (<http://opencv.org>)

**PYTHON:**

*Python is a widely used general-purpose, high-level programming language.* (<https://www.python.org/about/>)

**RFID TAGS:**

RFID tagging is an ID system that uses small radio frequency identification devices for identification and tracking purposes. (<http://whatis.techtarget.com/definition/RFID-tagging>)

**ROI:**

Region of Interest.

**ROS:**

The Robot Operating System (ROS) is a flexible framework for writing robot software. It is a collection of tools, libraries, and conventions that aim to simplify the task of creating complex and robust robot behaviour across a wide variety of robotic platforms. (<http://www.ros.org/about-ros/>)

**TOPICS:**

Topics are named buses over which nodes exchange messages. Topics have anonymous publish/subscribe semantics, which decouples the production of information from its consumption. (<http://wiki.ros.org/Topics>)

**UBUNTU:**

Ubuntu is a complete desktop Linux operating system, freely available with both community and professional support. (<https://www.ubuntu.com>)

## 2. Preface

### 2.1. Origin of the project

This research work begins as part of a larger-scale project called INHANDS (Interactive Robotics for Human Assistance in Domestic Scenarios) IBEC (The Institute for Bioengineering of Catalonia), which has as the aim to offer assistance in an interactive way to people with some degree of disability or elderly who suffer from limitations when it comes to carrying out daily activities in the kitchen.

The approach used to address this problem is to facilitate the user the execution of tasks by means of cooperation between human and robots through activity recognition and native commands like voice or gestures. For the execution of the project InHANDS, the department of Robotics has an automated kitchen prototype, several cameras that allow the complete visualization of the kitchen and robots for the execution of tasks.

As we mentioned before, InHANDS needs activity recognition among the multiple tasks that it will perform and it is precisely there where this project focuses on, providing by means of a Visual Perception System the object recognition, its position and most importantly, the activity recognition from its manipulation.

### 2.2. Motivation

Nowadays, one of the most interesting research areas in Computer Vision is the study of human activity. There are several fields of application in which human activity recognition is outstanding, e.g. video surveillance, accident prevention, assistance to disabled people. Nevertheless, the difficulty of conceptualizing parameters or relevant characteristics that define and distinguish human activities became a topic of research at the moment.

Aggarwal and Ryoo in their work "Human Activity Analysis: A Review" [1], carried out an interesting study on the different trends and theories to tackle the study of human activity. They distinguish between two big groups to classify the different existing approaches: Single - layered approaches and Hierarchical approaches. In addition, they contemplate another type of approaches: human-object interactions and group activities, being human-object interactions the ones that turn out to be the most attractive to deal with in our research with this type of analysis.

Aggarwal and Ryoo define the approach mentioned above in the following terms:

*"The most typical human-object interaction recognition approaches are the approaches ignoring interplays between object recognition and motion estimation. In those works, objects are generally recognized first, and activities involving them are recognized by analyzing the objects' motion. They have made the object recognition and motion estimation independent or made it so that the motion estimation is strictly dependent on the object recognition." [1]*

## 2.3. Related works

The reference number [1] is an important work that allows us to understand the methodological that we want to develop to accomplish Human Activity Recognition. As we mentioned before, they define the hierarchical approach-based taxonomy, which in its higher level consists of "single - layered approaches" and " hierarchical approaches". Regarding the first ones, they consider that they are appropriate for gestures and actions recognition by sequential characteristics. Hierarchical approaches are applied on human activity representation with high level of abstraction. These are described in terms of much simpler activities called "sub-events". If we get down a level in this branch we find three classes of approaches "Statistical", "Syntactic", and "Description-based". In Aggawar and Ryoo's words:

*"Statistical approaches construct statistical state-based models concatenated hierarchically (e.g. layered hidden Markov models) to represent and recognize high-level human activities.*

*Similarly, syntactic approaches use a grammar syntax such as stochastic context-free grammar (SCFG) to model sequential activities. Essentially they are modeling a high-level activity as a string of atomic-level activities.*

*Description-based approaches represent human activities by describing sub-events of the activities and their temporal, spatial, and logical structures." [1]*

In this context we will apply "Human - Object interactions" partially applying some characteristics of the "Syntactic approaches" and "Description-based" approaches.

"Video-based event recognition: activity representation and probabilistic recognition methods" [9] it is a work that is in line with the "Description-based" methodology, Its recognition system has two clearly differentiated modules, the first one is "Motion Detection and Tracking" and the second, "Event Analysis" . Being the first one of our interest due to the movement detection, that is one of the stages that we are interesting in implement, we agree on having a scene view provided by only one camera and segment by subtracting



the background, although we differ in the method . They do it based on intensity variations and we apply "Image Difference".

As for the works using the "Syntactic approaches" method we are interested in the one presented by Moore and Essa [10], in which they represent every "action event" with a unique symbol allowing to represent a sequence of interactions as a string of symbols. In our case the procedure differs in one aspect. Our symbol would turn into a word and an activity would be made of a list of words not necessarily in order.

The methodological alternative to recognition that we used is based on BOW (bag-of-words) which is widely explained in references [7] and [8]. Overall, BOW allows us to treat an image as a document in which we find words and their repetition in order to recognize the document, using "features" or "words". Liefeng and Sminchescu in their work "Efficient Match Kernels between Sets of Visual Features for Recognition" [7] state that BOW is one of the most popular methods to represent images, by being conceptually simple and computationally efficient. They support this using BOW together with several types of classifiers for three sets of databases, obtaining satisfactory results. Ryoo in "Human Activity Prediction: Early Recognition of Ongoing Activities from Streaming Videos" [8] considers an important objective the activity recognition before this activity finishes, that is to say during its execution. This way, a probabilistic prediction of these can be performed, which matches with the idea of this project: how we want to approach our activity recognition.

One of the most relevant works and in line with the aim of our research is the one presented by Jinna Lei, Xiaofeng Ren and Dieter Fox in "Fine-Grained Kitchen Activity Recognition using RGB-D" [2] where as its title indicates it fulfils Human activity recognition in a kitchen. One of the premises they consider in the work is to demonstrate the ability to identify objects using a Kinect-style camera as the main resource, adding that if a major robustness is wanted we could use it in combination with RFID tags.

The information about input data for the activity recognition they use divide into two basic categories:

- "1. Hand and object tracking, using depth to robustly track the positions of hands and objects, and detect when and where hands interact with the objects (e.g. grasp);*
- 2. Object and action recognition, using both depth (shape) and colour (appearance) to identify objects and to recognize the actions being performed on them."* [2]

The method they use when focusing on the actions in their project consists of:

*"7 common actions: place (PL), putting an object into the smart space from offscreen;*

*move (MV), moving an object inside the space; chop (CH), chopping vegetables and fruits; mixing (MX), stirring things and mixing them together; pouring (PR), pouring liquid or grain from one container to another; spooning (SP), using a spoon to transport stuff between containers; and scooping (SC), moving piles of stuff using hands."* [2]

They prove the reliability of that system defining the preparation of a cake as the activity to recognize. This activity is expressed in terms of 7 objects, 17 actions, about 6000 frames and approximately 200 seconds length.

Unlike the previous one, we try to fulfil our proposal considering actions in a simpler approach, so taking the objects in the scene and their movements. Therefore, our actions will be the followings:

**REMOVE:** It means that the user removed the object from the scene.

**ADD:** It means that the user added the object to the scene.

**UNCHANGED:** It means that the object is still present in the scene.

**MOVE:** It means that the user moved the object in the scene.

### 3. Introduction

This work proposes an idea to tackle human activity recognition while it is performed. This recognition is limited to the kitchen environment and basic activities, such as the ones related to the preparation of breakfast.

The object interaction approach takes into account the importance of these objects being in the field of vision, being brought to this one, removed or moved at the scene; actions carried out by a human being, therefore there is no need to make an analysis of the trajectories of its movements in this project.

#### 3.1. Objectives of the project

##### 3.1.1. General objective:

- To recognize the most probable human activity from a pre-established list in a domestic environment during its execution.

##### 3.1.2. Specific objectives:

- To objects on the basis of its colour.
- To the position of the recognized objects.
- To the activities our system will have to recognize.

#### 3.2. Scope of the project

Proactive assistance needs to recognize human activity while it is performed. This work will focus on the recognition of repetitive actions by taking into account the manipulated objects and their movements. To do so, it is needed to identify and locate the present objects in the scene by computer vision, and detect their position changes due to the user manipulation.

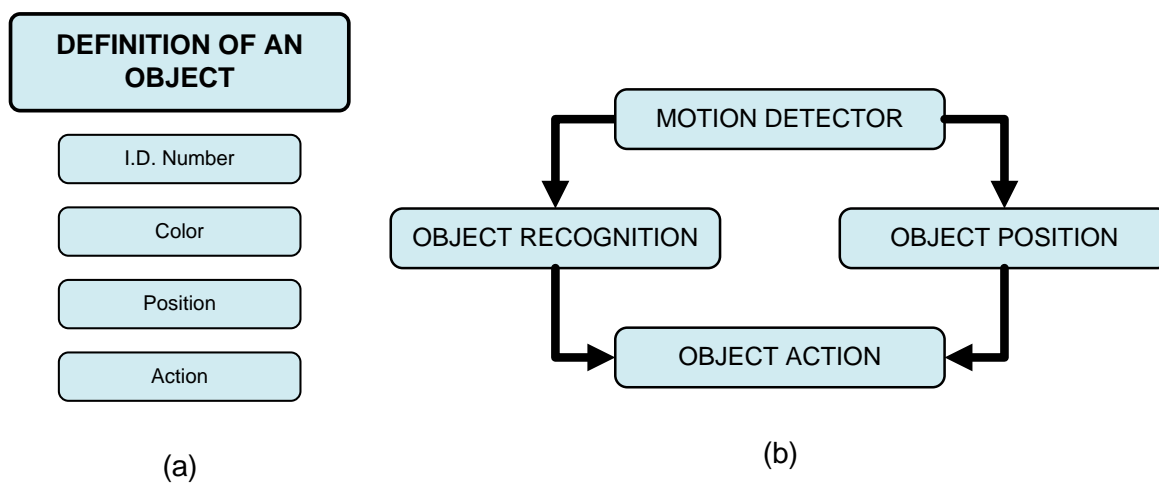
It is not pretended to continuously track the objects, but only register their initial and final positions. With the application of proactive assistance in mind, we'll look for methods capable of assigning probabilities to activities, describe an activity recognition nearly at the real time 0.25 seconds.

## 4. Conceptualization of an object

### 4.1. Definition of an object

The definition of an object in this work consists of 4 parameters or characteristics as illustrated in Fig. 4.1 (a) that are relevant in our proposal. These parameters are:

1. Identification number to be able to distinguish one from another of the same colour.
2. The colour that defines the model of the object recognized by our system.
3. The position that consists in the centroid coordinates based on the frame of reference specified through the homography.
4. The action that defines basically 4 options of object-manipulation by the user (Add, Remove, Move, Unchanged).



**Fig. 4.1.** (a) Graphical model of our definition of an object. (b) General flow chart of the definition of an object.

The functioning of the complete system starts with the definition of the object, which we can outline in 4 parts: the first one is a motion detector that allows segmenting and capturing the image, the three remaining parts consist of the search for those characteristics that we previously defined as relevant. The Fig. 4.2 (b) allows you to see that there is system processes performed in parallel, such as recognising the object, finding its location and finally establishing the action carried out by the user, which is something that depends on



the previous processes.

## 4.2. Requirements

We will explain the system requirements by dividing them into the hardware and software used for the functioning of the system.

### 4.2.1. Hardware



(a)



(b)



(c)

**Fig. 4.2.** (a) Kitchen used for the experimentation. (b) Computer Lenovo Y500. (c) Cameras and integrated sensors Microsoft Kinect.

Regarding the workspace for the tests and experimentation, we used the kitchen prototype provided by IBEC. As you can see in Fig. 4.2 (a), it has an aluminium structure on top for the movement of one of the robots and the installation of the cameras and projector. The camera (Kinect) we used is positioned in order to provide a zenithal view of the scene. On this one we only use its colour camera. When working with image processing the computer that we use must have a specific performance, like high processing speed (3rd generation Intel Core i7-3630QM 2.40GHz 1600MHz 6MB) and a graphic card that allows us to use simulators without any problem (NVIDIA GeForce GT750M GDDR5 2GB).



**Fig. 4.3.** Set of eleven objects for tests realized.

#### 4.2.2. Software

Based on the requirements of the project InHANDS from IBEC, we work using the operating system Linux 12.04 LTS and our framework ROS version HYDRO that gives us a perfect compatibility with the drivers from the different implemented cameras and a node structured system. In this way, the system turn into a ROS package with different nodes that communicates between them or to any element of the system through messages defined like topics.



**Fig. 4.4.** Logos of the used software (a) ROS, (b) OpenCV, (c) Python.

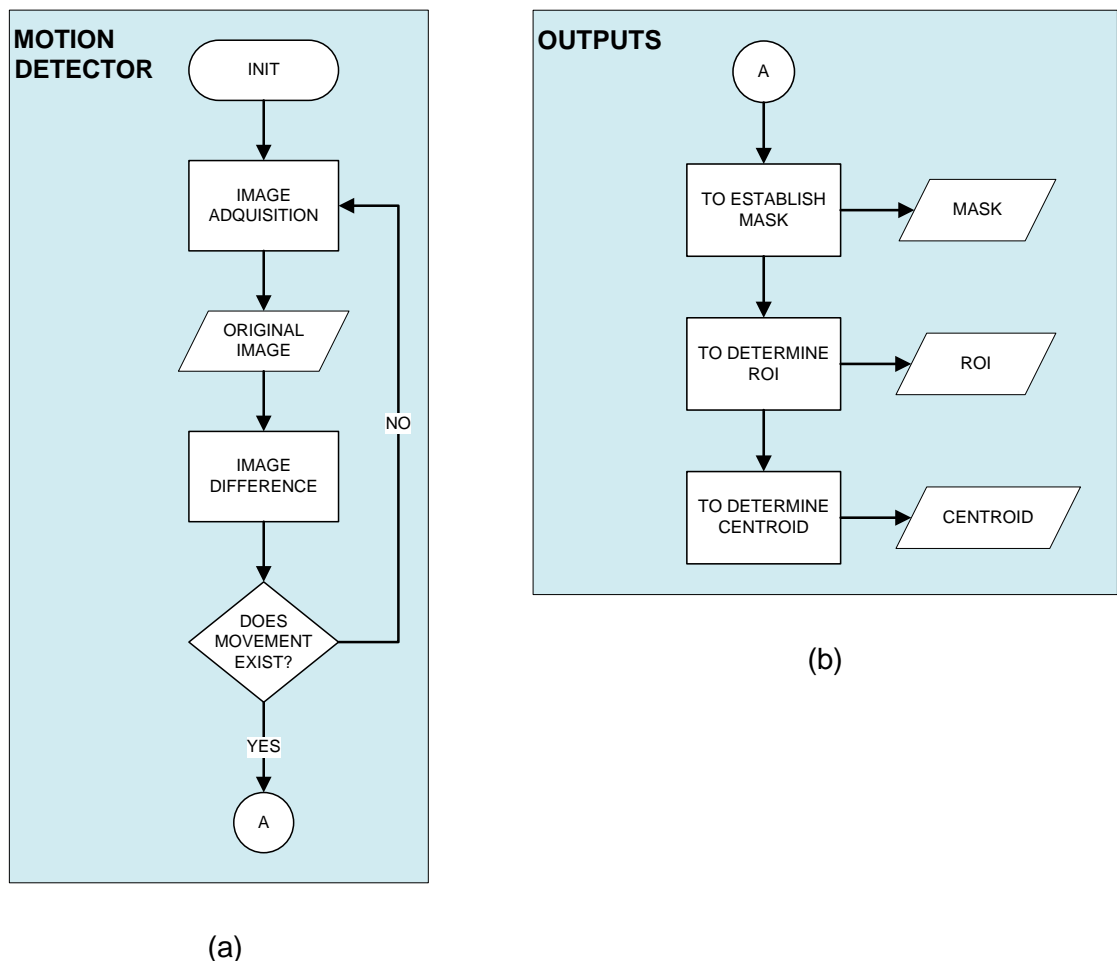
Since the process requires several tools and computer vision algorithms, we used OpenCV that as the name implies it is an open source software and therefore compatible with ROS, so it allows developing applications in Real Time. As for programming language we selected Python due to the simplicity when writing a code.



### 4.3. Motion detector

In any Computer Vision system segmentation is one of the crucial points so that in the following stages the process is carried out in a less complex way.

In this case we decided to deal with the image capturing by establishing motion thresholds, which will only proceed when a space in which the user has stopped generating movement in the scene is detected. As explained in the "Scope of the project" section, we are interested in capturing images that contain the objects in its initial and final position. This is based on two reasons: the first one is that we do not perform continuous tracking and the second one is that in this way we avoid the occlusion of the objects viewing angle.



**Fig. 4.5.** Flow charts: (a) Motion Detector, (b) Outputs from Motion Detector.

As demonstrated in the flow chart of Fig. 4.5, one of the most important methods applied in this part of the system is the image difference, specifically "The Mixture of Gaussian method" according to the reference [3] defined in the following terms:

*"First, the method maintains more than one model per pixel (that is, more than one running average). This way, if a background pixel fluctuates between, let's say, two values, two running average are then stored. A new pixel value will be declared as foreground only if it does not belong to any of the maintained models.*

*Second, not only is the running average maintained for each model, but also the running variance. This one is computed as follows:*

$$\sigma_t^2 = (1 - \alpha)\sigma_{t-1}^2 + \alpha(p_t - \mu_t)^2 \quad (\text{Ec. 4.1})$$

$p_t$  = Pixel value at a given time  $t$

$\mu_t$  = Average value

$\alpha$  = Learning rate

$\sigma_t^2$  = Variance

*The computed average and variance form a Gaussian model from which the probability of a given pixel value to belong to this Gaussian model can be estimated. This makes it easier to determine an appropriate threshold since it is now expressed as a probability rather than an absolute difference. Also, in areas where the background values have larger fluctuations, a greater difference will be required to declare a foreground object.*

*Finally, when a given Gaussian model is not hit sufficiently often, it is excluded as being part of the background model. Reciprocally, when a pixel value is found to be outside the currently maintained background models (that is it is a foreground pixel), a new Gaussian model is created. If in the future, if this new model becomes frequently hit, then it becomes associated with the background. "[3] [4]*

This image difference allows us to extract the objects from the background, which is dynamically updated while the system is working. First of all, it allows us to compare the



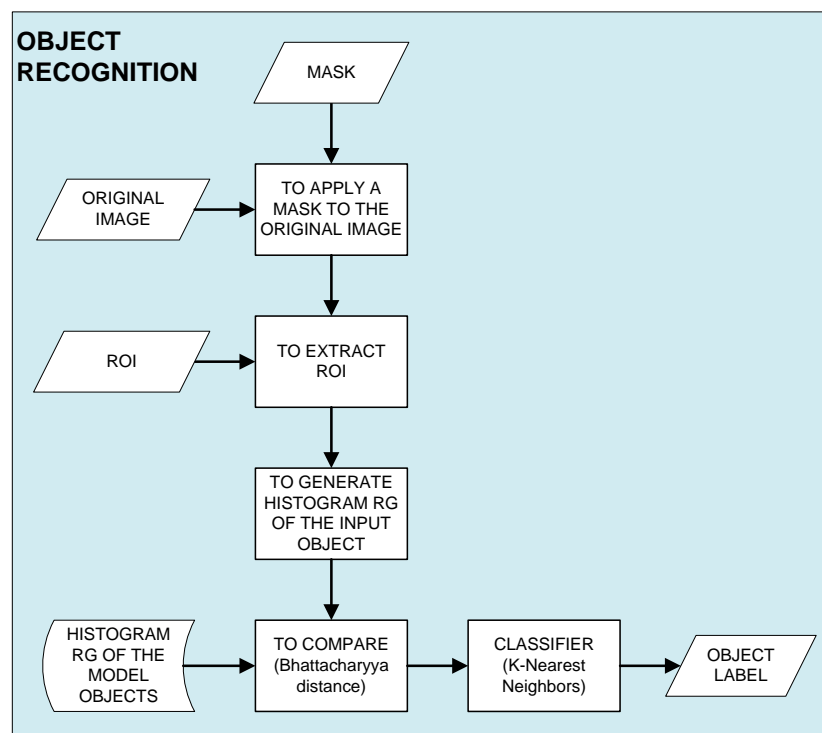


frame difference areas at time  $t$ ,  $t_1$ ,  $t_2$ . In case of being equal or minor to the established threshold we consider that there is no movement (Static). However, in case of overcoming that threshold we consider that there (Movement). Then the system performs a comparison of the frames  $t$ ,  $t_3$ ,  $t_6$ ,  $t_9$ ,  $t_{12}$ ,  $t_{16}$  to detect when the system returns to the state (Static) and to be able to process the new image of the scene.

Moreover, the image difference allows us to obtain a mask to carry out the segmentation of the regions of interest (ROI), which are probably the ones that contain the objects.

#### 4.4. Object recognition

For the object recognition, the performed process (Fig. 4.6) takes the mask obtained by image difference, applies it to the original image of the scene and extracts the region of interest. From each one of these is generated a histogram of 10 BINS in the **rg Chromaticity space** (Fig. 4.7). With this type of histogram we avoid problems related to the brightness variation in the scene. The colours black and white might cause problems to obtain the model histograms. To avoid this we normalize the images in RGB, imposing thresholds so the colours black or close to black are assigned to one only cell. We do the same procedure for the white colours.



**Fig. 4.6.** Flow chart: Object Recognition.

Once the histogram model has been obtained, we compare it with all the models stored in our database by means of Bhattacharyya distance (Ec. 4.2). There are 4 methods in OpenCV that allow comparison between histograms. In the tests (table 4.1) carried out with our system, Bhattacharyya distance gave us good results regarding percentage of mistakes and robustness in samples with noise. In addition, its computation time is not very long if we compare it to other methods.

$$d(H_1, H_2) = \sqrt{1 - \frac{1}{\sqrt{\tilde{H}_1 \tilde{H}_2 N^2}} \sum_I \sqrt{H_1(I) \cdot H_2(I)}} \quad (\text{Ec. 4.2})$$

$H_1$  = Histogram 1

$H_2$  = Histogram 2

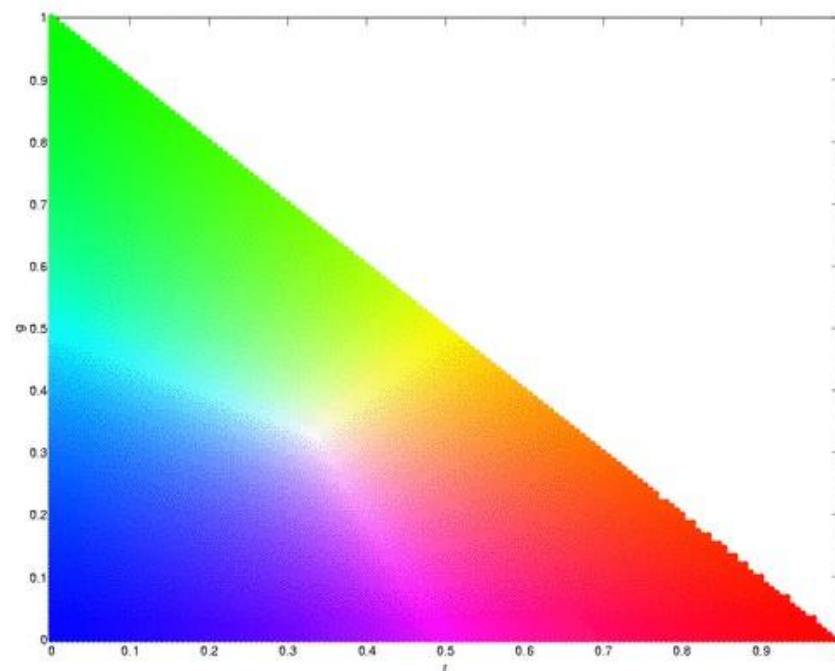
METHOD	Correlation	Chi-Square	Intersection	Bhattacharyya distance
<b>Samples</b>	100	100	100	100
<b>Error</b>	8%	8%	2%	2%
<b>Observations [5]</b>	Quick	Moderately Fast - More accurate matches.	Quick - and - dirty Matching.	Moderately Fast - More accurate matches.
<b>Range [Exact ... Mismatch]</b>	[1.0 ...-1.0]	[0.0 ...2.0]	[1.0 ...0.0]	[0.0 ...1.0]

Table. 4.1. Comparative of methods

To shape the objects in our database we take 5 different perspectives of the same object. This is done taking into account that some perspectives of the objects may lose the visibility of some of its typical colours and if we don't have these perspectives they could get confused with other objects with similar colours.

For the previously mentioned reasons, in our tests applying the classifier Knn (nearest neighbours) we had confusions using k=2 or 3, which did not happen using k = 1 because we ensure there is only one probable model.





**Fig. 4.7.** Normalized rg Colour Space. [8]

To evaluate our classifier we used the following confusion matrix [17] for all objects. The table 4.1 shows the results. We took 100 samples to get practice data 40 for validation and 30 for every model in the test, that is to say a total of 1870 images.

BOWL		PREDICTED LABEL	
KNOWN LABEL		Positive	Negative
	Positive	24	5
	Negative	0	1

Measure	Result
Precision	100,00%
Recall / Sensitivity	82,76%
Specificity	100,00%
Accuracy	82,76%

(a)

CEREAL		PREDICTED LABEL	
KNOWN LABEL		Positive	Negative
	Positive	13	7
	Negative	3	7

Measure	Result
Precision	81,25%
Recall / Sensitivity	65,00%
Specificity	70,00%
Accuracy	65,00%

(b)

CHOCOLATE		PREDICTED LABEL	
		Positive	Negative
KNOWN LABEL	Positive	20	4
	Negative	3	3

Measure	Result
Precision	86,96%
Recall / Sensitivity	83,33%
Specificity	50,00%
Accuracy	83,33%

(c)

COFFEE		PREDICTED LABEL	
		Positive	Negative
KNOWN LABEL	Positive	22	4
	Negative	2	2

Measure	Result
Precision	91,67%
Recall / Sensitivity	84,62%
Specificity	50,00%
Accuracy	84,62%

(d)

CUP		PREDICTED LABEL	
		Positive	Negative
KNOWN LABEL	Positive	22	6
	Negative	0	2

Measure	Result
Precision	100,00%
Recall / Sensitivity	78,57%
Specificity	100,00%
Accuracy	78,57%

(e)

GLASS		PREDICTED LABEL	
		Positive	Negative
KNOWN LABEL	Positive	24	5
	Negative	0	1

Measure	Result
Precision	100,00%
Recall / Sensitivity	82,76%
Specificity	100,00%
Accuracy	82,76%

(f)

JUICE		PREDICTED LABEL	
		Positive	Negative
KNOWN LABEL	Positive	18	5
	Negative	1	6

Measure	Result
Precision	94,74%
Recall / Sensitivity	78,26%
Specificity	85,72%
Accuracy	78,26%

(g)



MILK		PREDICTED LABEL	
		Positive	Negative
KNOWN LABEL	Positive	15	8
	Negative	2	5

(h)

Measure	Result
Precision	88,24%
Recall / Sensitivity	65,22%
Specificity	71,43%
Accuracy	65,22%

PLATE		PREDICTED LABEL	
		Positive	Negative
KNOWN LABEL	Positive	24	5
	Negative	0	1

(i)

Measure	Result
Precision	100,00%
Recall / Sensitivity	82,76%
Specificity	100,00%
Accuracy	82,76%

SPOON		PREDICTED LABEL	
		Positive	Negative
KNOWN LABEL	Positive	23	2
	Negative	0	5

(j)

Measure	Result
Precision	100,00%
Recall / Sensitivity	92,00%
Specificity	100,00%
Accuracy	92,00%

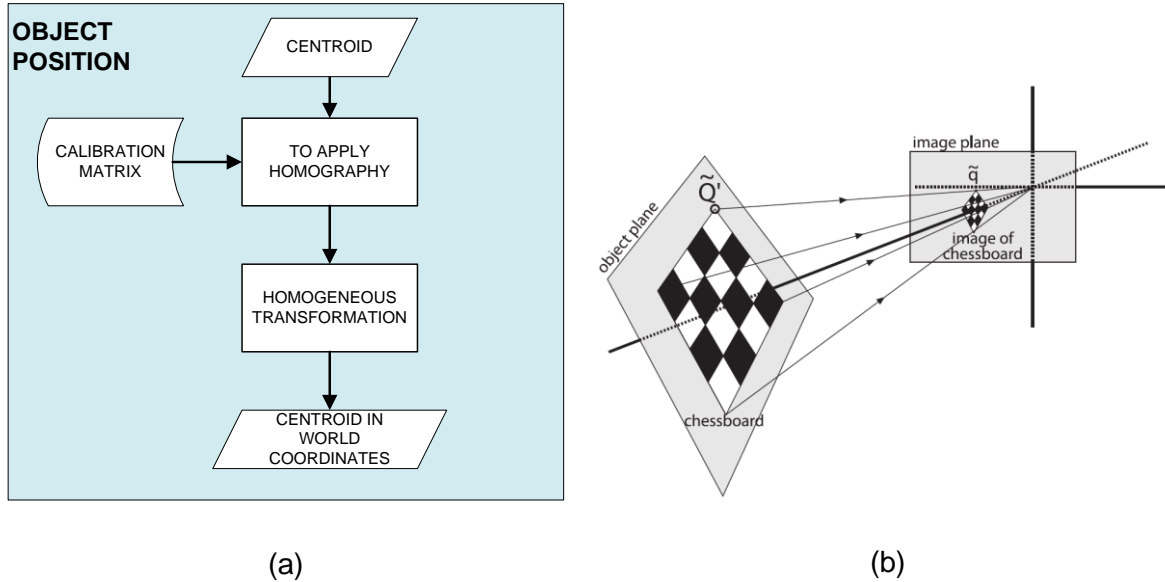
SUGAR		PREDICTED LABEL	
		Positive	Negative
KNOWN LABEL	Positive	16	10
	Negative	3	1

(k)

Measure	Result
Precision	84,21%
Recall / Sensitivity	61,54%
Specificity	25,00%
Accuracy	61,54%

Table 4.1. Confusion Matrix and measurements: (a) Bowl, (b) Cereal, (c) Chocolate, (d) Coffee, (e) Cup, (f) Glass, (g) Juice, (h) Milk, (i) Plate, (j) Spoon, (k) Sugar

## 4.5. Object position



**Fig. 4.8.** (a)Flow chart of Object Position, (b) View of a planar object as described by homography [5]

We obtain the object position that we have identified by calculating the centroid. However, this is not directly the useful value. This centroid is specified in a pixel that is to say with reference to the image coordinate system. Therefore the way to obtain its correspondence with the world coordinate system, which in this case the kitchen shelf, is to apply the homography, which we will express as matrix  $H$ . The following equations define the procedure to be followed:

$$\tilde{Q} = [X, Y, Z, 1]^T \quad (\text{Ec. 4.3}) [5]$$

$$\tilde{Q} = [X, Y, Z, 1]^T \quad (\text{Ec. 4.4}) [5]$$

$$\tilde{q} = sH\tilde{Q} \quad (\text{Ec. 4.6}) [5]$$



$Q$  = viewed point

$q$  = Point on the imager to which  $Q$  is mapped

$s$  = Scale factor

$H$  = Homography Matrix

Without loss of generality, we can choose to define the object plane so that  $Z = 0$ . [5]

$$\begin{bmatrix} \tilde{q} \end{bmatrix} = s \cdot \begin{bmatrix} \textit{Intrinsic} \\ \textit{Matrix} \end{bmatrix} \cdot \begin{bmatrix} \textit{Extrinsic} \\ \textit{Matrix} \end{bmatrix} \cdot \begin{bmatrix} \tilde{Q} \end{bmatrix} \quad (\text{Ec. 4.7}) [5]$$

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = s \cdot \begin{bmatrix} f_x & 0 & c_x & 0 \\ 0 & f_y & c_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} X \\ Y \\ 0 \\ 1 \end{bmatrix} \quad (\text{Ec. 4.8}) [5]$$

$f_x, f_y$  = Focal length

$c_x, c_y$  = Optical centres

$r$  = Rotation component

$t$  = Translation component

As you can see, before applying these equations it is necessary to obtain the intrinsic and extrinsic matrixes that were obtained following the procedure of camera calibration explained in reference [6]. Once applied the homography, we have the centroid in world coordinates expressed in millimetres. Then we apply the homogeneous transformation matrix to translate and rotate the coordinates centre to the point that we consider to be appropriate.

## 4.6. Object action definition

In this section we presented a different definition of "action" of the explained in [2], in this case human object interaction is described by basically 4 options of object-manipulation by the user (Add, Remove, Move, Unchanged).

Before carrying out the assignment of the last feature to define our object, we proceed to construct the object with the previously obtained characteristics (I.D. Number, Colour, and Centroid) and in "Action" we assign the state of "UNDETERMINED". To assign the state of "Action" correctly we need to fulfil a comparative analysis between two lists of objects; the first one in present time (t) and the second one in previous condition (state) (t-1).

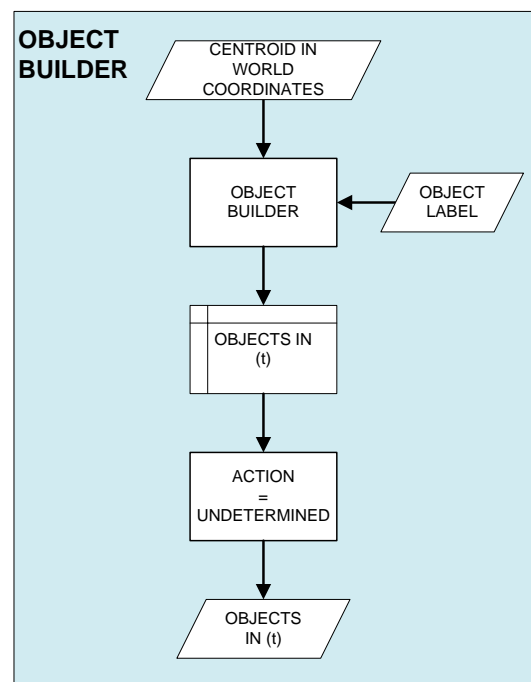
The possible actions are the following ones:

**REMOVE:** It means that the user removed the object from the scene.

**ADD:** It means that the user added the object to the scene.

**MOVE:** It means that the user moved the object in the scene.

**UNCHANGED:** It means that the object is still present in the scene.



**Fig. 4.9.** Flow charts of Object Builder.

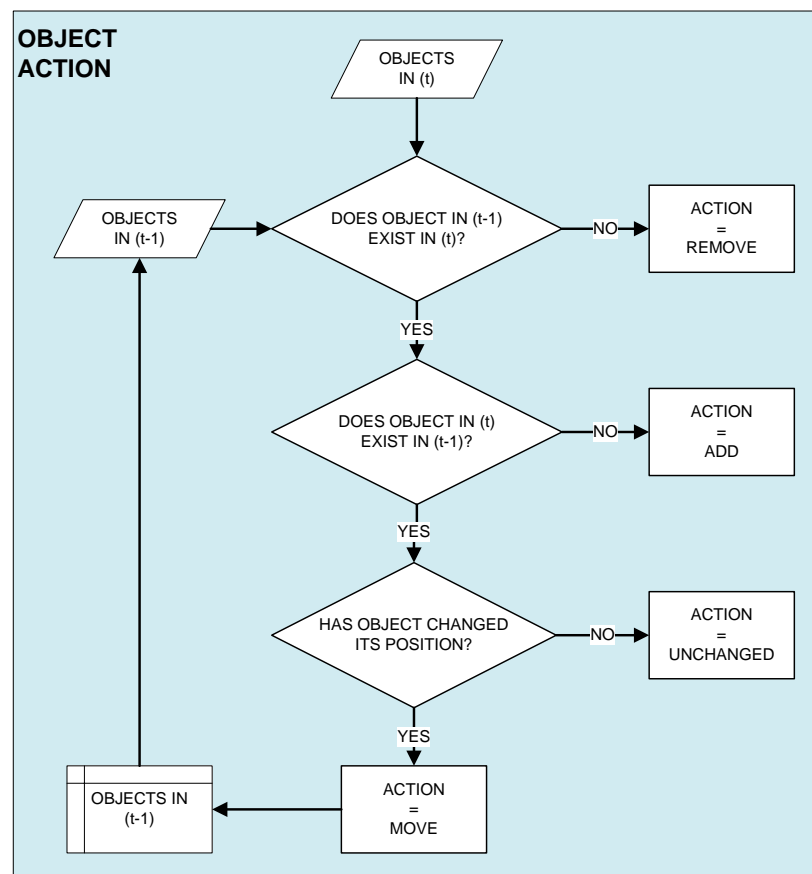




**REMOVE:** The first comparative is between the objects, therefore those that are present in ( $t_1$ ) and those not present in ( $t$ ) have been removed (action = REMOVE).

**ADD:** With the remaining elements in the lists now we check those objects in ( $t$ ) that are not present in ( $t_1$ ). These elements will be the objects recently added by the user (action = ADD).

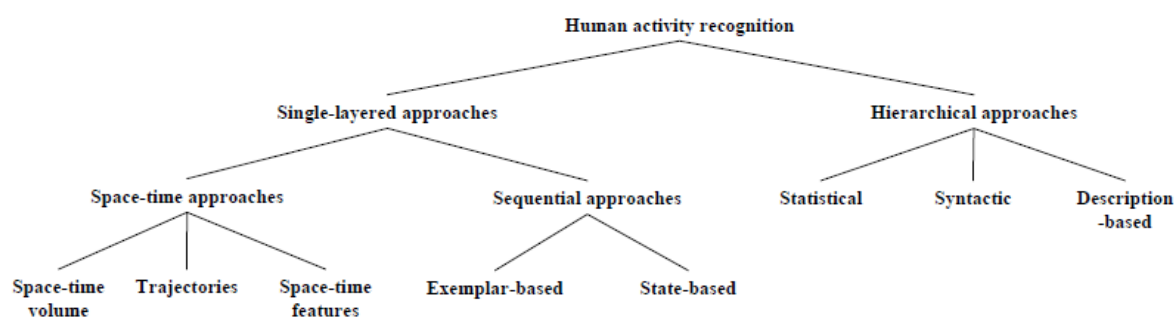
**MOVE AND UNCHANGED:** Now we have only the objects that coincide with the lists ( $t$ ) and ( $t_1$ ). We check the position of the objects, in other words, we compare its position in the list ( $t_1$ ) in relation to ( $t$ ). If this difference between positions exceeds a certain threshold, we consider that the user has moved the object (action = MOVE), whereas in the opposite case (action = UNCHANGED). It is important to have a small threshold that allows us to detect movement (= 5 mm.) for cases where the user takes the object and leaves it in the same position. To us it is registered as a movement.



**Fig. 4.10.** Flow charts Object Action.

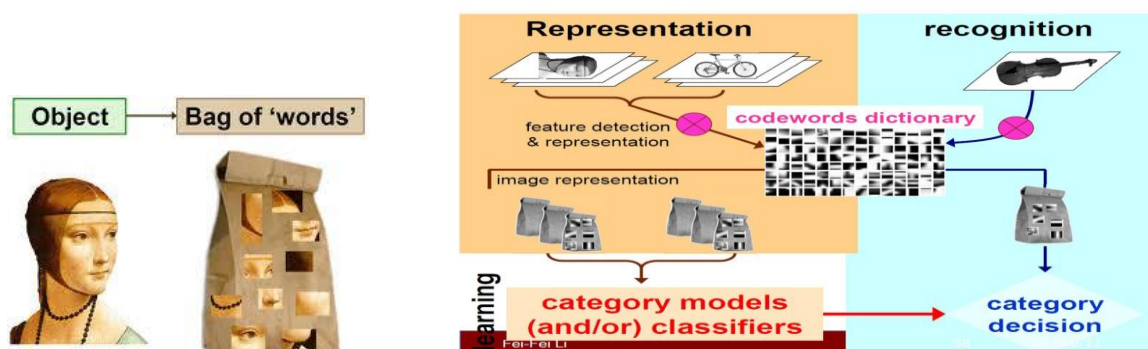
## 5. Human activity recognition

We want to initiate the activity recognition according to the classification proposed by Aggarwal and Ryoo [1]. It would be located inside the Hierarchical approaches methodology. It has many coincidences with the Syntactic approaches and Description-based approaches from the perspective of Human-Object Interactions. This proposal uses a syntax to define human activity as it does in Syntactic approaches. Nevertheless, we do not consider a sequential order. We consider sub-events from activities and its temporality but without the spatial consideration and a logical structure.



**Fig. 5.11.** The hierarchical approach-based taxonomy [1]

The methodology that comes closer to the implemented model is BOW (bag of words), "*BOW represents each local visual feature with the closest word and counts the occurrence frequencies in the image*" [7]. Then doing one comparative every object in the image with its characteristics it should represent a "word", and a specific set of words should represent an activity, it is necessary to stress that this set of words is not limited by a specific sequence of the words. The relevancy of each one of these words in a set would allow us to differ between activities.



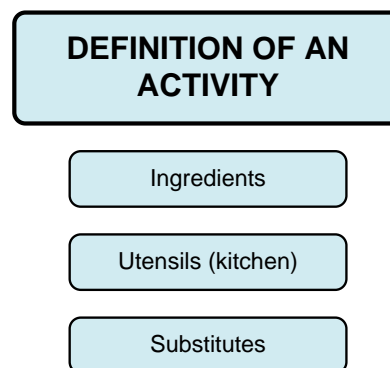
**Fig. 5.12.** BOW bag of words [11]



## 5.1. Definition of an activity

As explained in chapter 3, the four activities that were defined to appear in the tests for this system as well as the set of objects are related to breakfast. These activities are: the preparation of chocolate milk, coffee with milk, juice and cereal.

How to define an activity? This is one of the questions that arose during the project; in this case it is inspired by a recipe, so we will use **ingredients**, kitchen **utensils** and possible **substitutes** to define an activity.



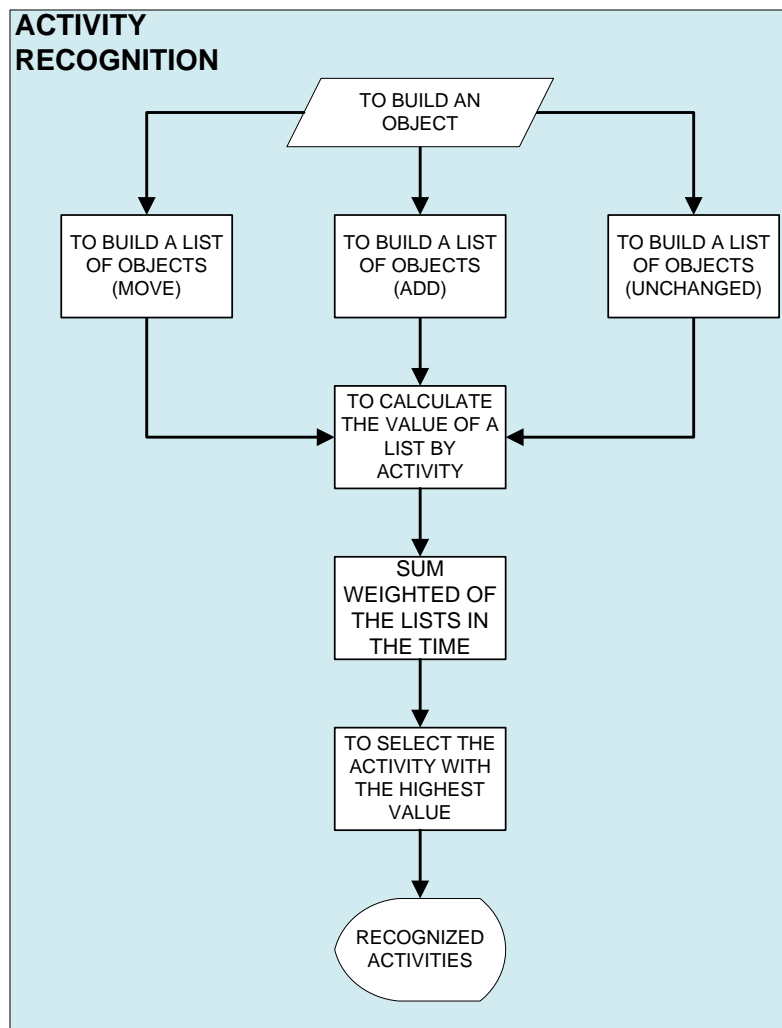
**Fig. 5.13.** Definition of an activity.

**Ingredients:** It is a list of ingredients related to the activity described, e.g. coffee-activity (Coffee, milk, sugar).

**Utensils:** It is a list of kitchen utensils related to the activity described, e.g. coffee-activity (cup, spoon)

**Substitutes:** It is a list of replacement for kitchen utensils or ingredients related to the activity described, e.g. coffee-activity (glass)

## 5.2. Distance function



**Fig. 5.14.** Flow charts Activity Recognition.

In chapter 4 we explained the whole process of object recognition. As a result of this process we obtain a message defined in ROS as `/collision_object`, in this format we include all the information that we need on the objects in the scene. With this information available as input for our process of activity recognition we create 3 lists with objects. Those lists were defined by the performed action by the user (MOVE, ADD, UNCHANGED).

Then we proceed with the calculation of the value of every list (list of Ingredients, list of utensils, list of substitutes), this value is calculated taking into account the contribution or relevancy of each one of these objects in an activity. We understand that the same list of objects will have a different value for each of the activities. The objects will be ingredients, utensils or substitutes depending on the activity.



$$\begin{bmatrix} Lv_{Act\_1} \\ \vdots \\ Lv_{Act\_N} \end{bmatrix}_{[M]} = a \cdot \begin{bmatrix} Ingd_{Act\_1} \\ \vdots \\ Ingd_{Act\_N} \end{bmatrix}_{[M]} + b \cdot \begin{bmatrix} Uts_{Act\_1} \\ \vdots \\ Uts_{Act\_N} \end{bmatrix}_{[M]} + c \cdot \begin{bmatrix} Subs_{Act\_1} \\ \vdots \\ Subs_{Act\_N} \end{bmatrix}_{[M]} \quad (\text{Ec. 5.9})$$

$$\begin{bmatrix} Lv_{Act\_1} \\ \vdots \\ Lv_{Act\_N} \end{bmatrix}_{[A]} = a \cdot \begin{bmatrix} Ingd_{Act\_1} \\ \vdots \\ Ingd_{Act\_N} \end{bmatrix}_{[A]} + b \cdot \begin{bmatrix} Uts_{Act\_1} \\ \vdots \\ Uts_{Act\_N} \end{bmatrix}_{[A]} + c \cdot \begin{bmatrix} Subs_{Act\_1} \\ \vdots \\ Subs_{Act\_N} \end{bmatrix}_{[A]} \quad (\text{Ec. 5.10})$$

$$\begin{bmatrix} Lv_{Act\_1} \\ \vdots \\ Lv_{Act\_N} \end{bmatrix}_{[U]} = a \cdot \begin{bmatrix} Ingd_{Act\_1} \\ \vdots \\ Ingd_{Act\_N} \end{bmatrix}_{[U]} + b \cdot \begin{bmatrix} Uts_{Act\_1} \\ \vdots \\ Uts_{Act\_N} \end{bmatrix}_{[U]} + c \cdot \begin{bmatrix} Subs_{Act\_1} \\ \vdots \\ Subs_{Act\_N} \end{bmatrix}_{[U]} \quad (\text{Ec. 5.11})$$

$Lv_{Act}$  = List Value by Activity

$Ingd_{Act}$  = percentage value based on the occurrence of the Ingredients by Activity

$Uts_{Act}$  = percentage value based on the occurrence of the Utensils by Activity

$Subs_{Act}$  = percentage value based on the occurrence of the Substitutes by Activity

$[M, A, U]$  = MOVE, ADD, UNCHANGED

$a, b, c$  = Constant,  $a + b + c = 1$

To obtain the probable activity that is being performed we add in the form of weighted the values obtained from every list by activity.

$$\begin{bmatrix} Act\_1 \\ \vdots \\ Act\_N \end{bmatrix} = \alpha \cdot \begin{bmatrix} Lv_{Act\_1} \\ \vdots \\ Lv_{Act\_N} \end{bmatrix}_{[M]} + \beta \cdot \begin{bmatrix} Lv_{Act\_1} \\ \vdots \\ Lv_{Act\_N} \end{bmatrix}_{[A]} + \gamma \cdot \begin{bmatrix} Lv_{Act\_1} \\ \vdots \\ Lv_{Act\_N} \end{bmatrix}_{[U]} \quad (\text{Ec. 5.12})$$

$Act$  = Activity

$\alpha, \beta, \gamma$  = Variables depending on the time,

$$\alpha = \frac{dec\_time}{2} \quad (\text{Ec. 5.13})$$

$$\beta = \frac{dec\_time}{2} \quad (\text{Ec. 5.14})$$

$$\gamma = (1 - dec\_time) + min\_cont \quad (\text{Ec. 5.15})$$

$$dec\_time = dec\_time_{t-1} + \frac{1}{average\_time \cdot frame\_rate} \quad (\text{Ec. 5.16})$$

$dec\_time$  = Value of decline in the weighting of the objects that they have not changed into the scene (unchanged)

$min\_cont$  = Value of minimum contribution of the objects that they have not changed into the scene (unchanged)

$average\_time$  = Average time for the execution of predefined activities

Based on the results obtained during the multiple tests Fig. 5.16 to 5.19 parts (a), as seen that the data results obtained in **instantaneous activity recognition** are not explicit, therefore was necessary filtering data results by means of the accumulative result in the time.

**Accumulative activity recognition**, finally the recognized activity is the result of the maximum resultant value of the sum of the samples of activity recognized instantaneously, that is to say one **value accumulated in one period of time**. The system begins the recognition when it detects movement and stops realizing it when this movement stops existing for a period prolonged in comparison to the normal time between movements inside an activity. The results are presented in Fig. 5.16 to 5.19 parts (b).

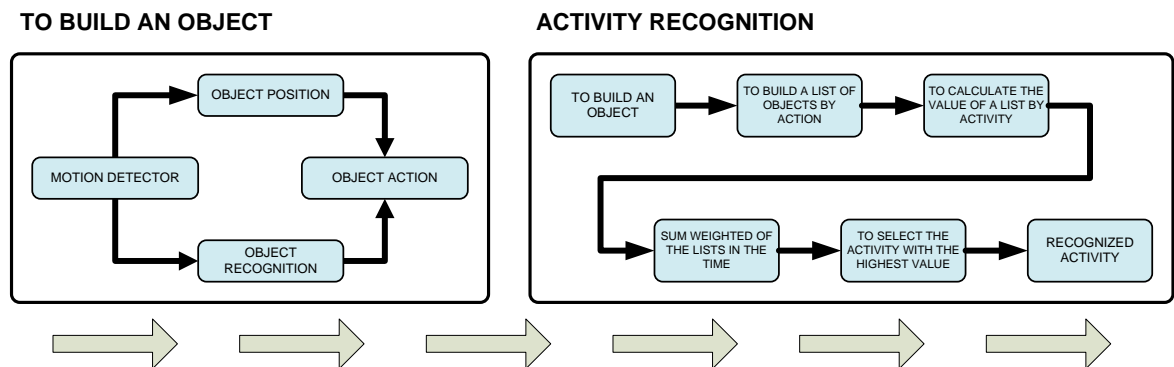


$$Activity\_recognized = \max \left\{ \sum_0^{T_{samples}} \begin{bmatrix} Act\_1 \\ \vdots \\ Act\_N \end{bmatrix} \right\} \quad (\text{Ec. 5.17})$$

$T_{samples}$  = Total samples of instantaneous activity recognized

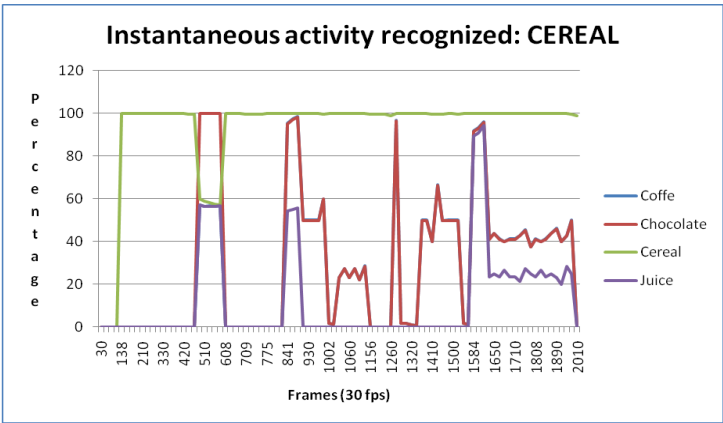
### 5.3. Results

The complete system that is obtained to perform our proposal of activity recognition is outlined in figure 5.15, each of its constitutive parts were explained in the chapter 4 and section 5.2.

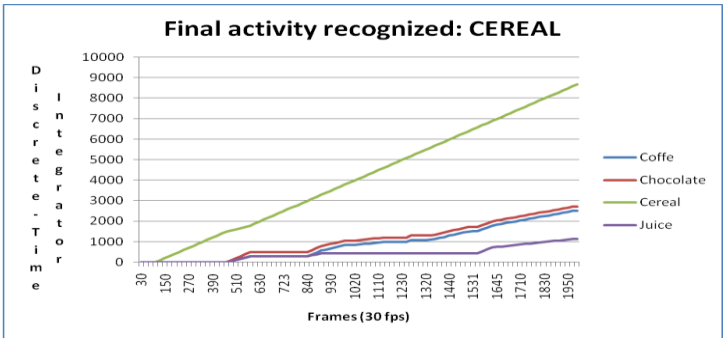


**Fig. 5.15.** Complete implemented system.

The graphs 5.16 to 5.19 expose a sample of the multiple tests that we carried out, in these examples the activities are isolated that is, without previous or posterior activities, (a) corresponds to the instantaneous recognition of activity and (b) corresponds to the recognition of activity accumulated, in other words, the activity performed in an interval of time during which there was movement.

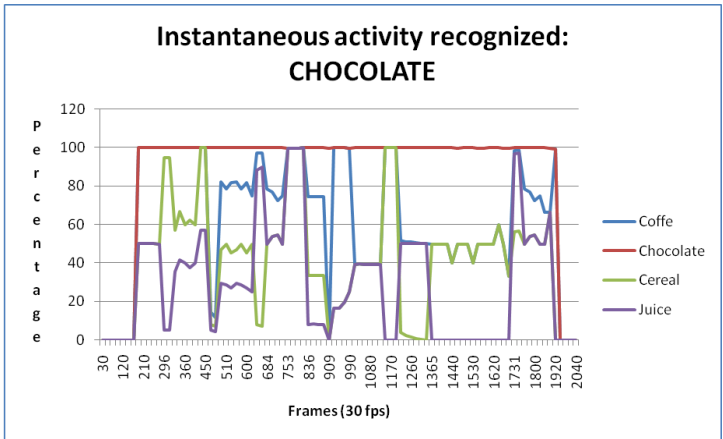


(a)



(b)

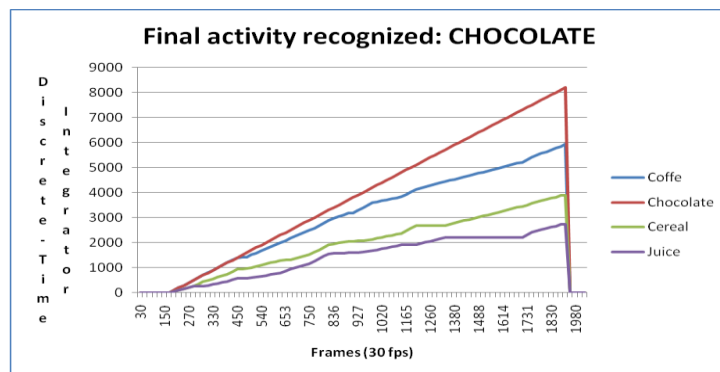
**Fig. 5.16\_** (a) Instantaneous activity recognized: CEREAL, (b) Final activity recognized: CEREAL



(a)

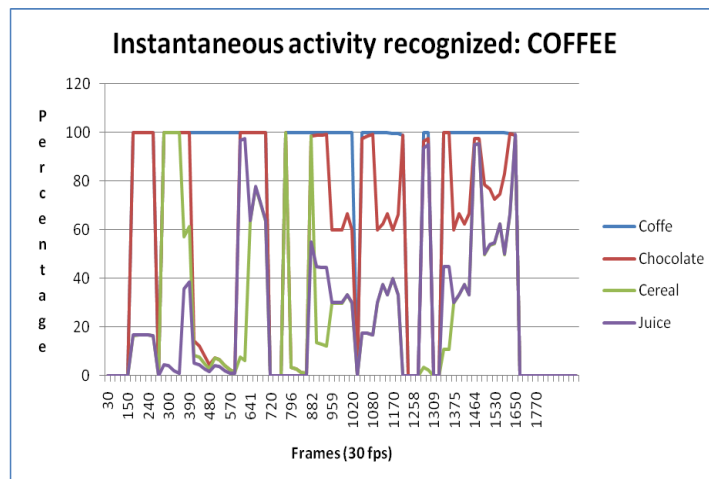




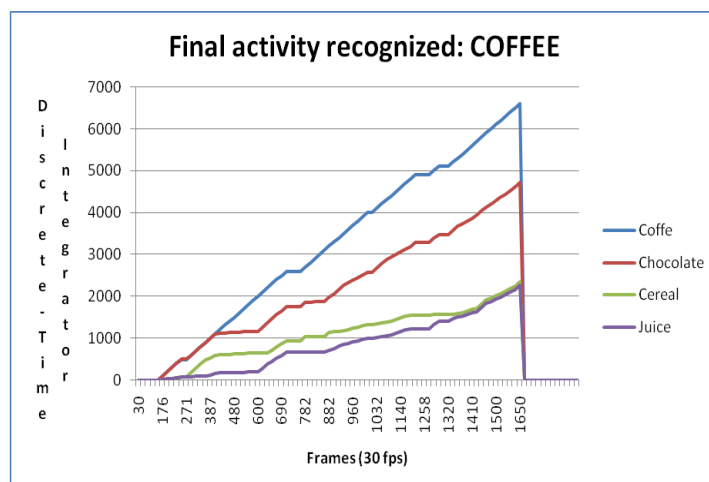


(b)

**Fig. 5.17.** (a) Instantaneous activity recognized: CHOCOLATE, (b) Final activity recognized: CHOCOLATE

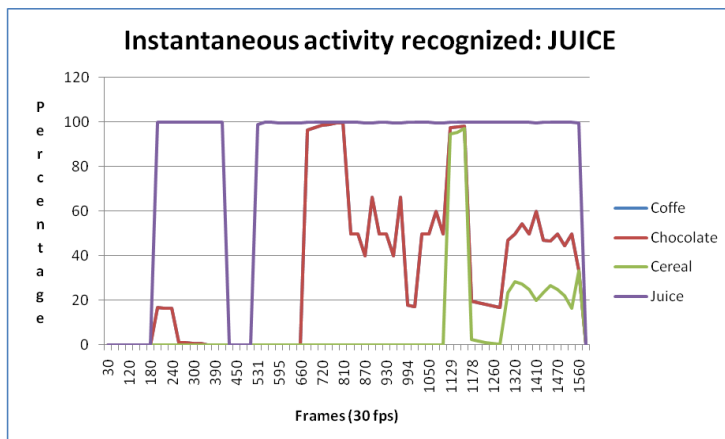


(a)

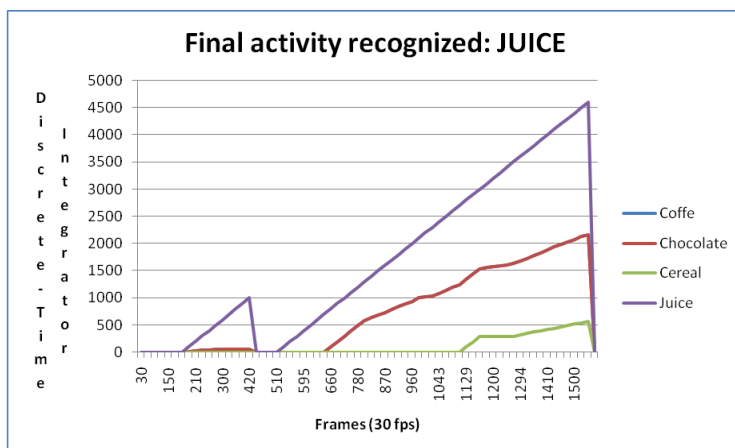


(b)

**Fig. 5.18.** (a) Instantaneous activity recognized: COFFEE, (b) Final activity recognized: COFFEE



(a)

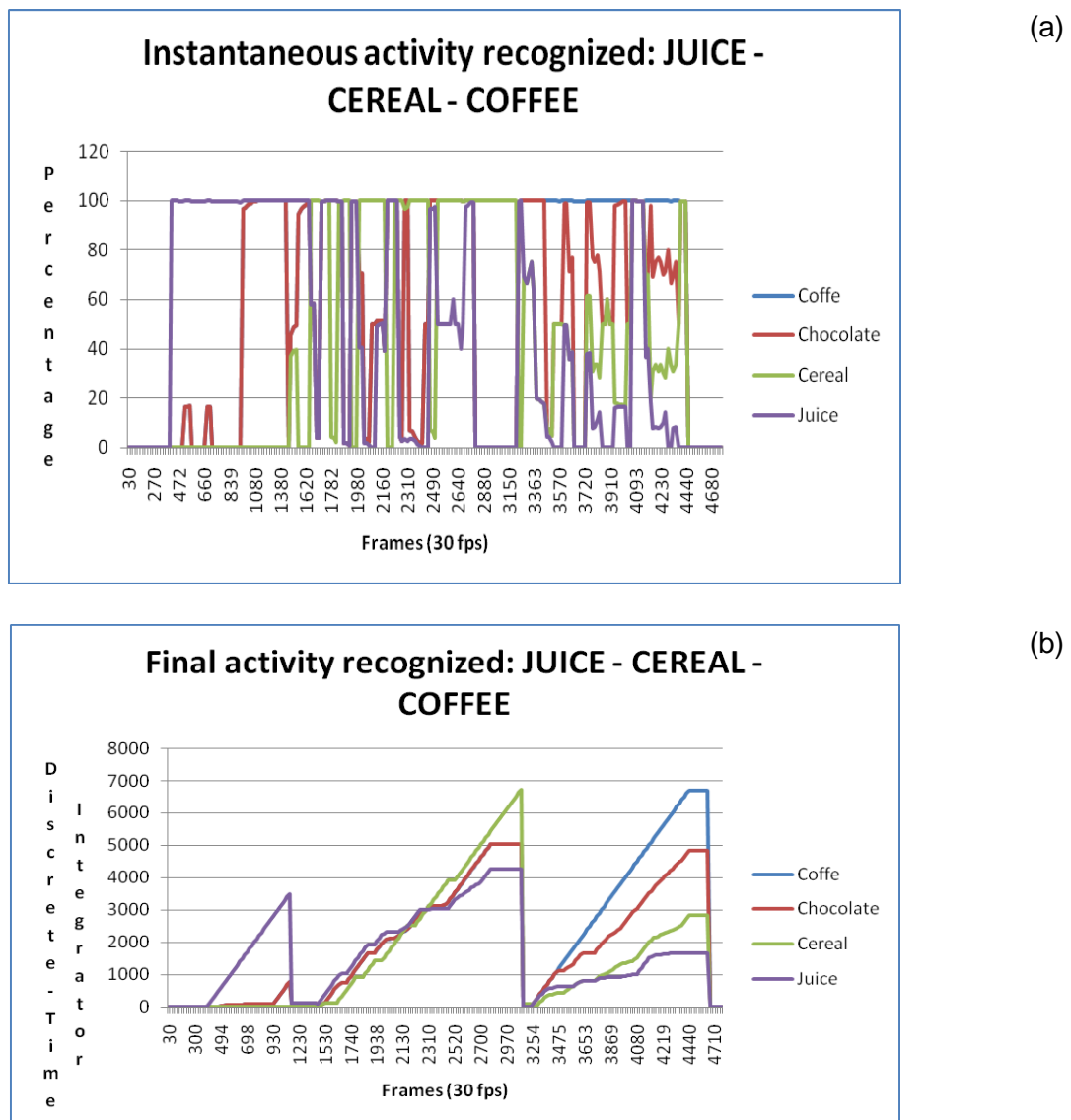


(b)

**Fig. 5.19.** (a) Instantaneous activity recognized: JUICE, (b) Final activity recognized: JUICE

The figure 5.20 illustrates the activity recognition in a constant way, with previous activities, posterior activities and including objects that do not intervene in the activity to evaluate the robustness of the system. We have to emphasize that in all the tests the recognition was fulfilled with occlusions, to allow completely natural movements by the user.





**Fig. 5.20.** (a) Instantaneous activity recognized: JUICE - CEREAL - COFFEE, (b) Final activity recognized: JUICE - CEREAL - COFFEE

Based on the figure 5.20 (b) illustrates the activities recognized, the first case is Juice-activity with correct response; the second case is Cereal-activity the result is not satisfactory due to human interaction with many objects in the initial frames until the middle of duration the activity that don't correspond at the activity for this reason the system have confusions; and the last activity is coffee the performance is satisfactory in the figure the evidence is clear.

The total time of execution of our algorithm taking as example 5 objects in the scene is

0.2529 seconds long.

		Parallel Execution				
	Motion Detector	Object Recognition	Object Position	Object Action	Activity Recognition	Total time
Seconds	0,0193	0,0433	0,003	0,1673	0,023	0,2529

Table 5.2. Total Time of execution by Node

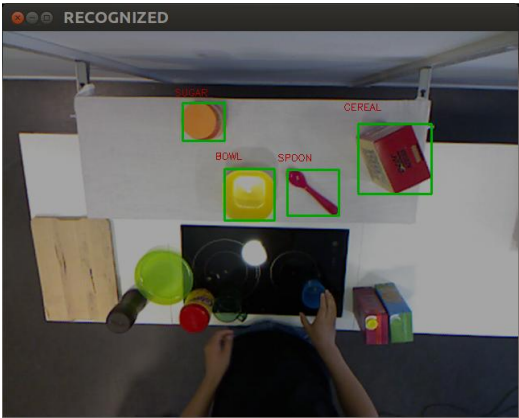


Fig. 5.21. (a) System object recognition graphic interface

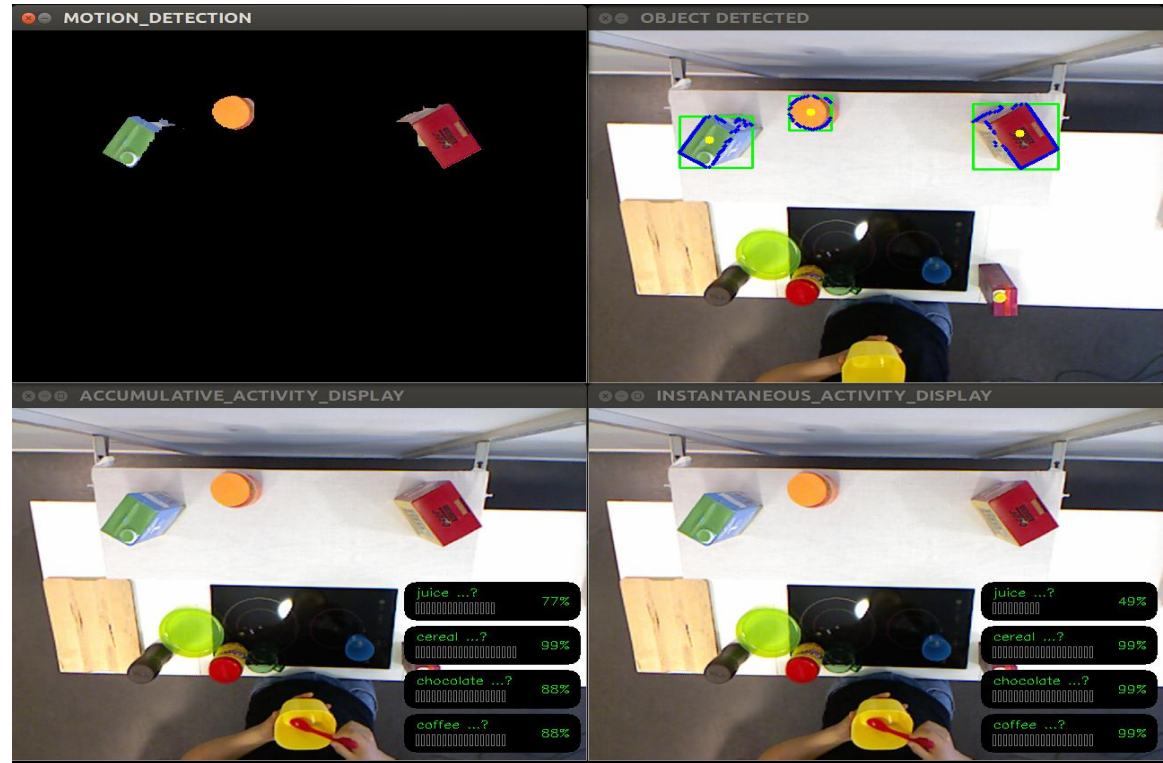


Fig. 5.22. (a) System activity recognition graphic interface



## Conclusions

In this work we have demonstrated that it is possible to recognize human activity in a simple way based on the interaction with objects which recognition is performed by means of computer vision techniques that *are not intrusive to the user*. In addition we achieve almost real time execution with an average time of 0.25 seconds approximately.

We have presented a new *definition of "action"* based on what happen with the objects under the assumption that they are only moved by the user. In this case human object interaction is described by basically 4 options of object-manipulation by the user (Add, Remove, Move, Unchanged).

For the recognition of the activity we have developed a simple structure inspired by a recipe. Hence, we have grouped objects in three classes: ingredients, utensils and possible substitutes. An activity is then defined by the presence of its pre-defined objects lists, demonstrating that it is applicable to the activity recognition process.

Our activity recognition system has been designed to work in a continuous way, without activity segmentation from the test video sequences. In order to evaluate the robustness of the system, these videos include activities previous and posterior to the activities selected, besides other objects that do not directly intervene. We have also to emphasize that in all the tests the recognition was fulfilled with occlusions, to allow completely natural movements from the user.

Our proposed method is able to overcome the common problems in computer vision, brightness and occlusion. The algorithm generally presents a trustworthy behaviour though these are present in some samples. Nevertheless, other activity recognition techniques might complete our project in order to offer higher confidence in the results, such as user movement recognition.

In addition, we have intentionally not established predetermined movements to recognize the activities. By doing so we can obtain a totally flexible and scalable system by just adding extra definitions in base of our structure for recognizing new activities.

A future interesting work would be to develop a statistical study to determine which is the relevancy of ingredients, utensils and substitutes for the different activities. The result would be useful to tune the algorithm and increase its robustness. Another improvement would be implementing a Learning algorithm in order to determine  $\alpha$ ,  $\beta$ ,  $\gamma$  in our distance function.



## Thanks

My gratefulness to IBEC for facilitating the use of your tools and resources for the elaboration of this thesis, especially to Dr. Joan Aranda Lopez and I.E. Emili Boronat.

To my teachers who have provided the necessary formation to me to fulfil this work.

To the government of my dear country Ecuador that has financed my studies.

To the director of the Master Dr. Cecilio Angulo Bahón, who always was ready to provide his guide and help to us during the whole program.

To my parents, family and fiancée that have supported me all the time to reach this aim.

To God.

# Bibliography

## Bibliographic references

- [1] AGGARWAL, J.K. AND RYOO, M.S. *Human activity analysis: A Review*. ACM Computing Surveys, Vol. 43, No. 3, Article 16, April 2011, p.1-43
- [2] LEI, J., REN, X. AND FOX, D. *Fine-grained kitchen activity recognition using RGB-D*, In Proc. UbiComp '12, AMC Press, September 2012, p. 208–211
- [3] LAGANIÈRE, R., *OpenCV 2 Computer Vision Application Programming Cookbook*, Birmingham: Packt Publishing Ltd., May 2011, p. 272 -277
- [4] STAUFFER C. AND GRIMSON W.E.L., *Adaptive background mixture models for real-time tracking*, *Proceedings IEEE Conf. on Computer Vision and Pattern Recognition*, pp. 246-252, 1999.
- [5] Bradski G. and Kaehler A., *Learning OpenCV*, O'Reilly Media, Inc., September 2008, p. 201 - 204, 384 - 404.
- [6] MORDVINTSEV A. and ABID K., *OpenCV-Python Tutorials: Camera Calibration and 3D Reconstruction*, [http://opencv-python-tutroals.readthedocs.org/en/latest/py\\_tutorials/py\\_calib3d/py\\_calibration/py\\_calibration.html#calibration](http://opencv-python-tutroals.readthedocs.org/en/latest/py_tutorials/py_calib3d/py_calibration/py_calibration.html#calibration), February - June 2014.
- [7] LIEFENG B., SMINCHESCU C., *Efficient Match Kernels between Sets of Features for Visual Recognition*, In Advances in Neural Information Processing Systems (NIPS), December 2009.
- [8] RYOO M. S., *Human Activity Prediction: Early Recognition of Ongoing Activities from Streaming Videos*, IEEE International Conference on Computer Vision (ICCV), Barcelona, Spain, November 2011.
- [9] HONGENG S., NEVATIA R., BREMOND F., *Video-based event recognition: activity representation and probabilistic recognition methods*, Computer Vision and Image Understanding 96, 2004, p. 129–162.
- [10] MOORE D. AND ESSA I., *Recognizing multitasked activities from video using stochastic context-free grammar*, Proc. 18th Nat. Conf. Artif. Intell., pp.770 -776 2002
- [11] FEI-FEI LI, *Quick Reviews on Object Recognition and Bag-of-Words (BoW) Models*, <http://sensblogs.wordpress.com/2011/08/23/quick-reviews-on-object-recognition-and-bag-of-words-bow-models-by-fei-fei-li/>, , June 2014.





## Complementary bibliography

- [12] R. PATRICK GOEBEL, *ROS By Example*, Vol. 1, February 2014
- [13] OPENCV DEV TEAM, *Opencv Documentation*, <http://docs.opencv.org/2.4/modules/refman.html>, February - June 2014.
- [14] CREATIVE COMMONS ATTRIBUTION, *Ros Tutorials*, <http://wiki.ros.org/ROS/Tutorials>, February - June 2014.
- [15] PYTHON SOFTWARE FOUNDATION, *The Python Tutorial*, <https://www.python.org/>, February - June 2014.
- [16] Dennis D., Tin C., and Marou R., *Color Image Segmentation*, <https://theiszm.wordpress.com/tag/color-segmentation/>, February - June 2014.
- [17] EISNER R., *Basic Evaluation Measures for Classifier Performance*, <http://webdocs.cs.ualberta.ca/~eisner/measures.html>, June 2014.